DUMPSBOSS.

ISTQB Certified Tester Advanced Level, Test Automation Engineering

iSQI CTAL-TAE

Version Demo

Total Demo Questions: 13

Total Premium Questions: 131

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Introduction and Objectives for Test Automation	9
Topic 2, Preparing for Test Automation	42
Topic 3, The Generic Test Automation Architecture	33
Topic 4, Deployment Risks and Contingencies	26
Topic 5, Test Automation Reporting and Metrics	15
Topic 6, Transitioning Manual Testing to an Automated Environment	6
Total	131

QUESTION NO: 1

A TAS has been in use for two years and contains custom libraries, test data generators, adapters, test scripts, execution configurations, and installation procedures. The organization is preparing a major upgrade of the operating system used by the test agents.

Which of the following artefacts should be placed under configuration management to support controlled maintenance of the TAS?

A. Custom libraries and SUT adapters

B. Test scripts and execution configurations

C. Test data generators and automated installation procedures

D. Only the test scripts, because other TAS artefacts are implementation details

E. Only the operating-system version, because it is the planned change

ANSWER: A B C

Explanation:

Custom libraries, adapters, test scripts, execution configurations, and automated installation procedures are all TAS artefacts that should be controlled. Their versions and relationships affect whether a particular automated execution can be recreated and whether a change to the operating system or another dependency has introduced an incompatibility. Test data generators should also be versioned because their behaviour can affect the inputs and states used by automated tests.

Configuration management provides identification of baselines, controlled changes, and traceability between the TAS components used for a run. It is not sufficient to control only the test scripts while leaving the framework, configuration, and deployment mechanisms unmanaged. See the ISTQB glossary entry for configuration management.

QUESTION NO: 2

Which of the following is NOT a technical design consideration for a TAA?

A. The number of users for the SUT

B. Availability of interfaces for the SUT to be testable

C. Standards and Legal requirements, e.g data privacy

D. Data used by the SUT, e.g configuration, users

ANSWER: A

Explanation:

The number of users for the SUT is the correct choice because it is primarily a business and operational characteristic of the system under test rather than a direct technical-design input for the Test Automation Architecture (TAA). A TAA defines the technical structure needed to create, execute, control, and maintain automated tests. Its design must address how automation components interact with the system, how tests obtain and manage data, and how the architecture complies with relevant constraints. These concerns directly affect the TAA's layers, adapters, libraries, configuration, test environments, and maintainability.

Although the anticipated number of users may be relevant when defining performance-test objectives or workload models, it does not by itself determine the technical design of the automation architecture. A test automation engineer may use user-volume information when implementing a particular performance-testing solution, but that is a test-design or non-functional testing input, not a general TAA technical-design consideration. The ISTQB Advanced Level Test Automation Engineering syllabus distinguishes architectural design concerns from the specific test conditions and workload characteristics that automated tests may later exercise. See the ISTQB Certified Tester Advanced Level Test Automation Engineer certification page and iSQI Test Automation Engineer information.

QUESTION NO: 3

You have been asked todeterminea TAS for a new release of a SUT, test should be automatedwherever. The new release will consist of 5 new interfaces and an amendment to 3 existing interfaces. The new and amended interface will be deliver incrementally in 3 sprints, each lasting 2 weeks.

What would be the BEST Test Automation Solution (TAS) design in this scenario?

A. Automate tests at both Component and System Level. Only do this automation once every interface has been fully developed or amended and manual testing has completed successfully.

B. Automate tests at one level only, System level. Use only the newly developed interfaces and do not create any customized interfaces/test hooks.

C. Automate the tests at two levels, Component and System level. Create customized hooks at Component level for interface not yet developed or amended. Only use the newly developed or amended interfaces to test at System level.

D. Automate tests at one level only, Component level. Create customized interface/test hooks for this level where the interface has not yet been developed or amended.

ANSWER: C D

Explanation:

Automate the tests at two levels, Component and System level. Create customized hooks at Component level for interface not yet developed or amended. Only use the newly developed or amended interfaces to test at System level. is the strongest solution for an incremental delivery schedule. Component-level automation can begin as soon as a component is available, while customized interfaces or hooks isolate the component from dependencies that have not yet been delivered or changed. This makes automated feedback available within each sprint rather than postponing meaningful testing until all interfaces are complete. System-level automation should use the genuine delivered interfaces, preserving realistic end-to-end coverage as each increment becomes integrated.

Automate tests at one level only, Component level. Create customized interface/test hooks for this level where the interface has not yet been developed or amended. also supports early, repeatable automated verification during the staged implementation. Test hooks are a recognized testability mechanism: they provide controlled access to, or substitutes for, dependencies so that automation can execute deterministically before the complete operational environment exists. The broader two-level approach remains preferable where system coverage is required, but component-level automated tests with suitable hooks are a valid TAS design for the incremental interfaces described. These choices align with the Test Automation Engineer certification’s focus on testability, maintainable automation, and integrating automation into iterative development; see ISTQB Test Automation Engineer and iSQI CTAL-TAE.

QUESTION NO: 4

A development team changes the version of a library used by the TAS to communicate with the SUT API. The existing automated tests still compile, but the library release notes describe changed default timeout and retry behaviour.

What is the BEST action before relying on the TAS results for a release decision?

A. Run representative tests exercising API communication, then progressively verify the wider suite

B. Assume the TAS is unaffected because the automated tests compile successfully

C. Update the library in production before executing any automated tests

D. Disable all tests that use the API until the next planned release

ANSWER: A

Explanation:

The best action is to execute representative automated tests that specifically exercise the affected API communication behaviour and then progressively confirm the wider suite. Successful compilation does not demonstrate that the TAS

behaves correctly at run time. Changed timeout and retry defaults can affect synchronization, response handling, execution duration, and the classification of failures.

A focused initial check provides rapid evidence that the updated dependency can establish connections, send requests, process responses, and report relevant errors correctly. Broader execution then provides confidence that the change has not introduced effects elsewhere in the suite. The dependency version and verification evidence should be controlled through configuration management. See the ISTQB glossary entry for configuration management.

QUESTION NO: 5

A team has automated a set of business rules using a data-driven approach. The same test procedure is executed with many combinations of input values and expected results stored in an external data source. A business-rule change affects the expected outcome for one group of input combinations.

Which of the following maintenance actions is MOST appropriate?

A. Update the affected expected results in the external test data source

B. Replace the data-driven tests with separate recorded scripts

C. Disable all test cases using the changed business rule

D. Move the expected results into comments in the automation code

ANSWER: A

Explanation:

Updating the affected test data is most appropriate when the automated procedure remains valid and only the expected outcomes for particular input combinations have changed. Data-driven scripting separates the reusable test logic from the data used to execute it. This makes it possible to maintain data variations centrally without duplicating or modifying the same procedural logic in multiple tests.

The change should first be analysed to confirm that the business rule has been implemented as intended. The affected expected results can then be updated under configuration management and the relevant automated tests re-executed. Replacing the data-driven approach with linear scripts or disabling the affected tests would unnecessarily reduce maintainability. See the ISTQB glossary entry for data-driven testing.

QUESTION NO: 6

A test automation team is selecting automated tests for the first increment of a regression suite. The available manual tests include stable, frequently executed business flows, one-off exploratory checks, tests requiring subjective visual judgement, and tests that rely on unpredictable data from an external production service.

Which of the following characteristics support selecting a manual test for early automation?

A. The test is executed frequently and covers stable functionality

B. The test has clear, objective expected results

C. The required test data and preconditions can be controlled

D. The test is a one-off exploratory investigation

E. The test verdict depends only on subjective visual judgement

ANSWER: A B C

Explanation:

Stable functionality that is executed frequently is a strong candidate for early automation because repeated execution can provide a return on the engineering and maintenance investment. Clear expected results and controllable test data also support dependable automated verdicts and repeatable execution. These characteristics reduce the risk that the resulting automated test will be unreliable or costly to maintain.

One-off exploratory checks and tests requiring subjective judgement are usually less suitable for early automation. Tests dependent on uncontrolled production data also require additional testability or environment measures before they can provide reliable automated feedback. Selection should consider business value, feasibility, execution frequency, and maintenance cost rather than one factor alone. See the ISTQB Test Automation Engineer certification page.

QUESTION NO: 7

A TAS uses a commercial test automation tool and the default logs generated by the inconsistent formats such as different types of messages (pass/fail steps, screenshots, warnings, etc.) To solve this issue some custom logging functions have been created from the test scripts, making it possible to log the different types of messages with the same format. However, this may cause a problem due to excessive size of the logs which can make it difficult to find the required information. Assume that all the default logs will be disabled when running the automated tests and that some tests will not generate excessively sized logs.

Which of the following represents the BEST suggestion for implementing the custom logging functions?

A. Implement the custom logging functions without saving timestamps

B. Implement the custom logging functions to support different levels of tracing

C. Implement the custom logging functions without saving stack traces

D. Implement the custom logging functions to redirect the logs to multiple files

ANSWER: B

Explanation:

Implement the custom logging functions to support different levels of tracing is the best suggestion because log levels provide a deliberate, configurable way to balance diagnostic value against log volume. The custom logger can classify routine execution details separately from warnings, errors, and critical failures. During normal automated executions, the configured level can retain concise outcome and failure information, making reports easier to scan and helping users find actionable results quickly. When a test fails intermittently or requires investigation, a more detailed tracing level can be enabled to capture the additional context needed for diagnosis without imposing that volume on every execution.

This approach also preserves a consistent custom format across message types while allowing the Test Automation Solution to adapt logging detail to the test suite, execution environment, and troubleshooting needs. Levels should be applied consistently by the logging functions and selected through configuration rather than by editing individual test scripts. This supports maintainability and makes logging behavior predictable across the automated testware. The ISTQB test-automation engineering syllabus identifies logging and reporting as essential capabilities of a test automation architecture, including the need to provide useful information for analysis and debugging. General logging guidance likewise recommends severity-based levels to control verbosity. See the ISTQB Test Automation Engineer certification page and Python Logging HOWTO.

QUESTION NO: 8

A TAS executes end-to-end tests against a SUT that exchanges messages asynchronously with an external notification service. For most automated tests, the notification service is not the subject of the test and is not available in the test environment.

Which of the following are appropriate uses of a test double in this scenario?

A. Simulate defined service responses, including failures and timeouts

B. Capture messages sent by the SUT for later verification

C. Provide a controllable service state for each automated test

D. Remove the need to perform any integration testing with the real service

E. Eliminate the need for expected results in the automated tests

ANSWER: A B C

Explanation:

A controllable test double can simulate the notification service and return defined responses, including error and timeout conditions. It can also capture messages sent by the SUT, allowing the automated test to verify the content and timing of the interaction without relying on the unavailable external service. These uses improve controllability, observability, and repeatability of the automated tests.

A test double should not replace verification against the real service in every situation. Appropriate integration tests are still needed to validate the actual interface and contractual behavior when the real dependency is available. A test double is also not a substitute for defining expected results. The TAS must still determine whether the observed behaviour is correct. See the ISTQB Test Automation Engineer certification page.

QUESTION NO: 9

You are reviewing the testability of your SUT.

Which of the following BEST refers to the characteristic of OBSERVABILITY?

A. The ability of the SUT to perform its intended function for a specified period of time B. The ability to exercise the SUT by entering inputs, triggering events and invoking methods

B. The ability of the SUT to prevent unauthorized access to its components or data.

C. The ability to identify states, outputs, intermediate result and error messages in the SUT

ANSWER: C

Explanation:

The ability to identify states, outputs, intermediate result and error messages in the SUT correctly describes observability. In test automation engineering, observability is a testability characteristic describing the extent to which a tester or automated test can determine what the system is doing internally and externally. A system is observable when it exposes sufficiently useful evidence of its behavior, such as accessible state information, returned or displayed outputs, intermediate processing results, diagnostic logs, notifications, and meaningful error messages.

This visibility enables an automated test to establish whether the SUT reached the expected state and to accurately determine the result of a test execution. It also supports efficient diagnosis when a failure occurs, because the available information helps distinguish a product defect from an environmental, test-data, or automation issue. Observability therefore improves both the reliability of automated verdicts and the maintainability of the automation solution. The CTAL Test Automation Engineering syllabus treats controllability and observability as important testability considerations when designing or assessing automation support. See the ISTQB Test Automation Engineering certification page and the ISTQB glossary entry for observability.

QUESTION NO: 10

A TAS that performs automated testing in a single test environment was successfully manually installed and configured from a central repository, with all its components in the correct versions. It was also verified that all TAS components in this environment are capable of providing reliable and repeatable performance. The TAS will be used to run several suites of automated regression test scripts on various SUTs in the test environment. Your current goal is to complete all preliminary verifications to ensure that the TAS works correctly. Which of the following activities would you perform FIRST?

A. Create scripts to automatically install and configure the TAS in the test environment from the central repository

B. Check whether the TAS connectivity to all required internal systems, external systems, and interfaces is available

C. Run a given suite multiple times using TAS to determine whether all regression test scripts always provide the same result

D. Check whether all regression test scripts in a given suite have expected results

ANSWER: B

Explanation:

Check whether the TAS connectivity to all required internal systems, external systems, and interfaces is available is the first appropriate activity. The scenario already establishes that the TAS has been installed with the required component versions and that its individual components can perform reliably and repeatably. Before meaningful end-to-end automated test execution can begin, however, the installed TAS must be shown to communicate with every dependency needed during execution. This includes the relevant SUTs and interfaces, as well as supporting services such as identity providers, databases, test-data sources, messaging infrastructure, browser or device services, and externally hosted services where applicable.

Connectivity verification is an essential deployment and operational-readiness check because a TAS may be correctly installed while still being unable to execute tests due to unavailable endpoints, network routing, firewall rules, invalid credentials, certificates, permissions, or interface configuration. Establishing this access first provides a dependable basis for subsequent checks of test-script behavior and suite execution. It confirms that observed execution results can be attributed to the automated tests and SUT behavior rather than to an inaccessible environmental dependency. This aligns with the Test Automation Engineering syllabus focus on deploying, operating, and maintaining the test automation solution and its required interfaces. See the ISTQB Test Automation Engineer certification page and the ISTQB certification overview.

QUESTION NO: 11

Which of the following statements BEST describe aspects of the SUT to consider when designing a TAA?

A. All the interaction between SUT and TAS should be logged with the highest level of detail

B. All the internal test interfaces of the SUT should be removed prior to the product release

C. All the interfaces of the SUT affected by the tests should be controllable by the TAA

ANSWER: C

Explanation:

All the interfaces of the SUT affected by the tests should be controllable by the TAA is correct because controllability is a central testability characteristic to evaluate when designing a Test Automation Architecture (TAA). The architecture must enable the test automation solution to establish the required preconditions, provide inputs, invoke the relevant SUT functions, and manage the environment sufficiently to execute automated tests reliably. This does not necessarily mean controlling every interface exposed by the product; it concerns the interfaces that the automated tests need to use or influence.

Considering controllability early is important because it determines whether automated tests can perform repeatable actions without impractical manual intervention. For example, the TAA may need supported mechanisms to configure test data, reset an application state, simulate dependent services, or drive a user, service, and/or API interface. These mechanisms should be designed with the SUT's architecture, constraints, and intended test levels in mind. The ISTQB Test Automation Engineering syllabus identifies SUT testability, including controllability and observability, as an important architectural consideration for automation. See the ISTQB Test Automation Engineer certification page and iSQI's CTAL-TAE certification information.

QUESTION NO: 12

A project plans to migrate its GUI automation solution to a new browser engine. The test definitions are largely independent of the current browser, but the test adaptation layer contains browser-specific drivers, object-location mechanisms, and synchronization functions.

Which of the following activities are appropriate during the migration?

A. Update or replace the browser-specific adapters and drivers

B. Execute a representative compatibility suite against the new browser engine

C. Review object-identification and synchronization behavior

D. Maintain the relevant adapter versions under configuration management

E. Rewrite all test definitions regardless of their abstraction from the browser

ANSWER: A B C D

Explanation:

Updating or replacing the browser-specific adapters and drivers is appropriate because these components are responsible for the technical interaction with the SUT. Executing a representative compatibility suite validates that the revised adaptation mechanisms can correctly control and observe the application. Reviewing object-identification and synchronization behavior is necessary because browser differences can affect element availability, rendering, event timing, and supported automation interfaces. Maintaining both the old and new adapter versions under configuration management supports a controlled transition and rollback if required.

Rewriting all test definitions is not normally required when their abstraction from the browser-specific adaptation layer has been maintained. See the ISTQB Test Automation Engineering certification page.

QUESTION NO: 13

A project uses a keyword-driven TAS. Several teams contribute new keywords, and analysis shows that keywords with similar names perform different actions and some keywords contain direct references to specific test data.

Which of the following actions would improve the maintainability of the keyword library?

A. Define naming conventions and clear responsibilities for keywords

B. Consolidate overlapping keywords where their intended behaviour is the same

C. Parameterize variable test data used by keywords

D. Embed test-specific account identifiers in each keyword implementation

E. Allow duplicate keyword names when they are created by different teams

ANSWER: A B C

Explanation:

Defining keyword naming conventions and clear functional responsibilities improves the usability and consistency of the library. Reviewing and consolidating overlapping keywords reduces duplication and makes it easier for test authors to select the intended business-level operation. Parameterizing variable input data keeps the keyword reusable across different tests and separates test data from implementation.

Documentation of keyword purpose, parameters, and expected outcomes supports correct use by multiple teams. Embedding more test-specific data in keywords or allowing unrestricted duplicate names would increase coupling and maintenance effort. See the ISTQB glossary entry for keyword-driven testing.